CLAIMS:

1. A host for executing an operation requested by a requestor, the host comprising a memory configured to store information identifying the requestor and information related to the operation requested by the requestor, and processing means for determining the presence of the requestor and for freeing resources associated with that operation if the requestor is not present.

2. The host of claim 1, wherein the said memory stores information identifying the requestor as an object, wherein said processing means is configured to call a destructor to destroy the object unless a refresh signal is received within a predetermined time from the requestor.

3. The host of claim 2, wherein the host comprises means to send a signal to the requestor confirming receipt of the refresh signal.

4. The host of claim 2, wherein said memory is arranged to store a plurality of objects corresponding to a plurality of operations requested by the requestor, each object being linked to an object corresponding to a registration request from the requestor.

5. The host of claim 2, wherein the processing means is configured to execute operations requested by a plurality of requestors.

6. The host of claim 2, wherein the memory is configured to store an object corresponding to a registration request from each of the requestors and each subsequent operation requested by a requestor is linked to the object corresponding to its registration request.

7. The host of claim 4, wherein the processing means is adapted to free the resources associated with an object which is linked to an object corresponding to a registration request which has been deleted.

8. The host of claim 2, wherein an object comprises status information for allowing the status of the operation to be determined from the object.

9. The host of claim 2, wherein said memory is adapted to store an object for file open, read, write, seek and close operations.

10. The host of claim 2, wherein said memory is adapted to store an object for running a thread in the host.

11. The host of claim 2, wherein said processing means is adapted to send a signal to the requestor indicating the status of the operation, before the processing means calls a function to free resources associated with that object.

12. The host of claim 1, wherein the processing means is configured to store data concerning the status of the operation requested by the requestor which will remain in the memory of the host after the resources associated with the operator have been freed.

13. The host of claim 1, including connection means for connecting the host to another host via a network.

14. The host of claim 13, wherein said connection means is adapted for connection to a packet switched network.

15. The host of claim 14, wherein said connection means is adapted for connection to an asynchronous network.

16. A network comprising a first device connected to a second device across an asynchronous network, the first device being configured to execute an operation in response to a request from the second device to determine the presence of the second device and to free resources associated with the operator if the presence of the second device cannot be determined by the first device.

17. A method of executing an operation which is performed on a host remote from the requestor of the operation, the host and requestor being connected over an asynchronous network, the method comprising:
the requestor sending a signal to the host to request an operation to be performed by the host; and
the requestor intermittently sending a refresh signal from the requestor to the host in order to confirm that the requestor is still connected to the host.

18. The method of claim 17, wherein the requestor waits to receive a response from the remote host to indicate that the operation was successful.

19. The method of claim 18, wherein the requestor resends the signal requesting an operation if a response is not received within a predetermined time from the host.

20. The method of claim 17, wherein the requestor receives a signal from the host in response to sending a refresh signal.

21. The method of claim 20, wherein the requestor saves information relating to the status of the requested operation if no response is received from the host within a predetermined time.

22. The method of claim 17, wherein the requestor requests a registration operation with the host.

23. The method of claim 17, further comprising:
the requestor at least partially encrypting the request sent to the host.

24. The method of claim 23, wherein the requestor encrypts the request using information which uniquely identifies the requestor.

25. The method of claim 23, wherein the requestor encrypts the request using information which identifies an application running on the requestor which requires the request to be sent to the host.

26. The method of claim 23, wherein the requestor encrypts the request using the time when the request is encrypted.

27. The method of claim 23, wherein the requestor sends information to the host which will allow the host to decrypt the request with the encrypted request.

28. The method of claim 17, wherein the requestor sends its http address to the host.

29. A method of performing an operation in a host in response to a request from a requestor, the method comprising:
the host receiving a request from the requestor to perform an operation at the host;
storing information corresponding to the identity of the requestor at the host in the form of an object; and
freeing resources associated with that object if a refresh signal is not received by said host after a predetermined time.

30. The method of claim 29, wherein the host sends a signal concerning the status of the operation to the requestor if a refresh signal is not received within a predetermined time.

31. The method of claim 29, wherein the host stores data concerning the status of the operation which will remain after the resources associated with the operation have been freed.

32. The method of claim 29, wherein the host receives a registration request from the second device.

33. The method of claim 32, wherein the host decrypts the registration request and sends a signal to the requestor to indicate if the request has been successfully decrypted.

34. The method of claim 33, wherein the host uses information received from the host to decrypt the request.

35. The method of claim 33, wherein the host uses information which can be used to uniquely identify the host to decrypt the request.

36. The method of claim 33, wherein the host uses information which can be used to identify an application running on the requestor to decrypt the request.

37. The method of claim 33, wherein the host uses the current time the request was sent from the requestor to decrypt the request.

38. A host for executing a plurality of tasks in response to at least one request from a requestor, the host comprising a memory configured to store information identifying the requestor and information concerning the tasks such that information concerning the tasks can be preserved between tasks.

39. The host of claim 38, wherein information concerning the identity of the requestor is stored in the form of a registration object in the memory of the host

and wherein status information concerning each of the tasks is also stored in the form of objects in the memory of the host.

40. The host of claim 39, wherein the host comprises processing means configured to free resources associated with an object if the host does not receive a refresh signal from the requestor within a predetermined time.

41. The host of claim 40, wherein the processing means is configured to delete a registration object if a refresh signal is not received in time.

42. The host of claim 41, wherein the memory is configured to store each of the objects relating to tasks requested by a requestor linked to the registration object of that requestor, the processing means being configured to free the resources associated with each of the tasks requested by the requestor upon deletion of the registration object.

43. A network comprising a first device connected to a second device across an asynchronous network, the first device being configured to execute a plurality of operations in response to a request from the second device to determine the presence of the second device and to delete the operations and free resources if the presence of the second device cannot be determined by the first device.

44. A device configured to retrieve data from at least one second device and to output said data, the device being configured to intermittently check for the presence of the second device such that the device can determine if data can be retrieved from the second device.

45. The device of claim 44, wherein the device comprises a web server and is configured to publish data from the second device in the form of web pages.

46. The device of claim 45, configured to indicate that web pages formed from information from the second device are only available if the first device determines that the second device is present.

47. The device of claim 44, wherein the device includes storage means for storing a flag which indicates if the second device is present, and means for setting said flag to indicate that the second device is not present if a signal is not received from the second device within a pre-determined time.

48. A device configured to receive information from at least one second device, wherein the device includes storage means for storing a flag which indicates if at least one of the other second devices is present, and means for determining if the second device is capable of sending information.

49. A code module for controlling a local processor connected to one or more remote processors controlled by like code modules over a network, the code module being adapted to control the processor to communicate with a remote processor and to save information concerning a session formed between the local and remote processors.

50. The code module of claim 49, adapted to control the local processor to save information identifying the code module of a remote processor with which it has formed a session in the form of a registration object.

51. The code module of claim 50, adapted to control the local processor to delete the object and free the resources associated with that object once the session is finished.

52. The code module of claim 51, wherein the object requires a signal from the code module of the remote processor after a predetermined time otherwise the session is finished.

53. The code module of claim 49, adapted to control the local processor to manage a plurality of tasks requested by the remote code module in a single session and to store information concerning the tasks requested in a single session such that information from previously performed tasks is available for use by later tasks requested in the same session.

54. The code module of claim 49, adapted to control the local processor to store information related to each requested task in the form of an object, wherein each object associated with a task is linked to the registration object.

55. The code module of claim 54, adapted to control the local processor to store the object relating to a task until the registration object has been deleted even if the task has been completed.

56. The code module of claim 54, adapted to control the local processor to delete an object associated with a task if the registration object to which it is linked has been or is going to be deleted.

57. The code module of claim 49, adapted to control the local processor to send a signal to the remote processor confirming that the session is to be terminated prior to terminating the session.

58. The code module of claim 49, adapted to control the local processor to store status information after the session has been terminated.

59. The code module of claim 58, adapted to control the local processor to set a flag to indicate whether or not status information should be stored in the event that the session is terminated.

60. The code module of claim 49, adapted to control the local processor to store a plurality of registration objects each associated with a different remote code module.

61. The code module of claim 60, adapted to control the local processor to link each of the objects relating to a task requested by a remote code module with the registration object of that control module.

62. The code module of claim 49, adapted to control the local processor to send information requesting a session with a remote processor which is managed by a similar code module.

63. The code module of claim 62, adapted to control the local processor to send information uniquely identifying the processor which is controlled by the code module to the remote processor when requesting a session.

64. The code module of claim 62, adapted to control the local processor to send information identifying itself to the remote processor when requesting a session.

65. The code module of claim 62, adapted to control the local processor to repetitively send a refresh signal to the remote processor to prevent the remote processor from ending the session.

66. The code module of claim 49, adapted to control the local processor to store status information relating to a requested task a signal is received indicating that the session formed with the remote processor is about to be terminated by the remote processor.

67. The code module of claim 49, wherein the code module is adapted to interface with an application which commands the code module to instruct tasks on a remote processor.

68. The code module of claim 67, adapted to interface with an application which can perform client and/or server functions

69. The code module of claim 49, wherein the code module is adapted to interface to an encryption package which can encrypt or decrypt signals received from the code module.

70. The code module of claim 49, wherein the code module is adapted to control the local processor to encrypt signals sent from said code module to a remote processor and decrypt signals received from a remote processor.

71. The code module of claim 69, wherein the code module is adapted to control the local processor to encrypt the signals using information which uniquely identifies the local processor.

72. The code module of claim 69, wherein the code module is adapted to control the local processor to encrypt the signals using information identifying the code module.

73. The code module of claim 69, wherein the code module is adapted to control the local processor to encrypt the signals using a time relating to when an operation was controlled by the code module.

74. The code module of claim 49, wherein the code module is adapted to directly interface to the socket layer of the processor.

75. The code module of claim 74, adapted to control the local processor to switch sockets during a session.

76. The code module of claim 49, adapted to control the local processor to publish pages to the internet.

77. The code module of claim 49, further comprising a user interface.

78. The code module of claim 77, wherein the user interface comprises a webserver.

79. The code module of claim 49, adapted to control the local processor to store information about a plurality of further code modules all adapted to control the same processor.

80. The code module of claim 49, adapted to control the local processor to store the socket numbers used by each of the plurality of further code modules.

81. The code module of claim 79, adapted to control the local processor to store information for each of the further plurality of code modules concerning the applications which are interfaces to each of the plurality of further code modules.

82. The code module of claim 79, adapted to interface to the same application as at least one of the further plurality of code modules.

83. A primary code module for controlling a processor to store information concerning at least one secondary code module wherein the at least one secondary code module is adapted to interface to a predetermined application and to manage a session formed between said processor and a remote processor.

84. The primary code module of claim 83, adapted to control the local processor to store information concerning the socket used by the at least one secondary module.

85. The primary code module of claim 83, adapted to control the local processor to store information concerning the services which can be provided by said at least one secondary code module by virtue of the application to which it is attached.

86. The primary code module of claim 83, adapted to control the local processor to store information concerning a plurality of secondary code modules interfaced to a plurality of applications.

87. The primary code module of claim 83, adapted to control the local processor to store information concerning a plurality of secondary code modules interfaced to the same application.

88. The primary code module of claim 83, adapted to interface to an application.

89. The primary code module of claim 88, adapted to interface to the same application as at least one of the secondary code modules.

90. The primary code module of claim 83, adapted to control the local processor to listen on a predetermined socket which is known to all other authorised processors which are networked to the said processor.

91. The primary code module of claim 90, adapted to control the local processor to send said information in response to an enquiry from a remote processor such that said remote code module can contact one of the secondary code modules controlling said processor.

92. The code module of claim 49, adapted to be downloaded from a first device to a second device.

93. The code module of claim 92, adapted such that it can be downloaded by a second device which does not have such a code module.

94. A carrier medium carrying the processor readable code module in accordance with claim 49.

95. A plurality of devices each comprising a code module in accordance with claim 49.

96. The plurality of devices of claim 95, wherein each of the code modules is adapted to interface with an application which operates as a client and/or a server.

97. A method of sharing data files between a number of devices, the files being stored in a distributed manner amongst any number of the devices, wherein metadata containing at least the location of the files provided to be read by any of the devices with the appropriate access rights.

98. The method according of claim 97, wherein the metadata is stored on all of the devices.

99. The method of claim 97, wherein the metadata is stored on a registry device which does not store the shared files.

100. The method of claim 99, wherein each of the devices which can store a file registers information including at least the location of the file with the registry.

101. The method of claim 99, wherein the registry stores information concerning the attributes of the different types of files.

102. The method of claim 97, wherein the registry comprises processing means to distinguish between files with different attributes such that the registry can supply information concerning the just files with a specific attribute.

103. The method of claim 102, wherein the files relate to advertisements, the registry storing the information concerning the type of advertisement as well as its location, the registry also being able to distinguish between different types of advertisements such that the locations of advertisements of the same type can be retained in response to an enquiry requesting a certain type of advertisement.

104. The method of claim 102, wherein an enquiry is sent to the registry from a remote device requesting files having certain attributes and wherein the location of each of the files with the requested attributes is sent to the remote device.

105. The method of claim 102, wherein an enquiry is sent to the registry from a remote device requesting files having certain attributes and wherein the registry contacts each of devices where files are located with the requested attributes and these files are sent to the remote device.

106. The method of claim 97, wherein each of the devices comprises a code module in accordance with claim 49, to manage the sessions formed between the devices.

107. A method of transferring files from a first device to a second device, the method comprising:
the first device interrogating a registry device with which the second device will register if it is able to receive the file;
the first device sending a request to register with said second device and to establish a session if the second device is able to receive the file; and
sending the file to the second device.

108. The method of claim 107, wherein the first device sends the file to the registry if the second device is not able to accept the file.

109. The method of claim 107, adapted to send e-mail from the first device to the second device.

110. The method of claim 107, wherein the first device and the second device each comprise a processor which is controlled by a code module in accordance with claim 49.

111. The method of claim 110, wherein the code module of the first device and the code module of the second device negotiate with one another to switch sockets during transfer of the file.

112. A method of transferring a file from a first device to a second device, wherein the first device subdivides the file to be transferred into portions and wherein the sockets are changed between the transmission of individual portions.

113. The method of claim 112, wherein each portion contains information which indicates the socket of the following portion.

114. The method of claim 111, wherein the first device subdivides the file into portions of different sizes.

115. The method of claim 112, wherein the first device and the second device each comprise a processor which is controlled by a code module in accordance with claim 49.

116. A communication method between a first device and a second device, comprising the steps of:
the second device receiving a registration request from the first device, the registration request comprising information uniquely identifying the first device.

117. The method of claim 116, further comprising the step of the second device performing checks to see if the registration request came from a device to which the second device can provide services.

118. The method of claim 117, wherein the second device is adapted to perform a step in a sales transaction requested by the first device.

119. The method of claim 118, wherein the second device is adapted to refuse the transaction request of the first device, if the first device has been used in a previous fraudulent transaction.

120. The method of claim 116, wherein the first device and the second device each comprise a processor which is controlled by a code module in accordance with claim 49.

121. A plurality of devices connected over a common network, each device being able to access data files on at least one of the other devices, wherein at least one of the devices has a web server responsive to authorised access requests to output selected data files as webpages.

122. The plurality of devices of claim 121, wherein each device comprises a processor having a code module in accordance with claim 49.

123. A method of leasing file content to a remote device, the method comprising the steps of:
the lessor receiving a request from a remote device indicating that the device wishes to lease at least some file content,
the lessor receiving information identifying the file content to be leased, information uniquely identifying the device requesting the lease and indicating the required extent of the lease;

the lessor packaging the requested file content with information uniquely identifying the user and indicating the extent of the lease to form a packaged file; and

executing the packaged file such that the file content is delivered to the remote device.

124. The method of claim 123, wherein the requested file content is packaged with code which allow usage of the file by the remote device to be monitored by the lessor.

125. The method of claim 123, wherein the packaged file is encrypted.

126. The method of claim 123, wherein the packaged file is sent to the remote device and the remote device executes the packaged file.

127. The method of claim 123, wherein the packaged file is executed in a distributed manner between the lessor and the remote device.

128. The method of claim 127, wherein the file content is streamed to the remote device.

129. The method of claim 127, wherein the code which allow usage of the file by the remote device to be monitored by the lessor is executed remote from the remote device.

130. The method of claim 123, wherein the file is packaged with a user identifier, the user identifier comprising information which uniquely identifies the remote device, information which identifies the requested file content and information which identifies a code module running on the remote device which communicates with the lessor.

131. The method of claim 130, wherein the user identifier comprises temporal information.

132. The method of claim 123, further comprising the step of the lessor receiving a registration request from the user.

133. The method of claim 131, further comprising the step of the lessor receiving a registration request from the user.

134. The method according of claim 133, wherein the temporal information relates to the time and date when the user registered with the lessor.

135. The method of claim 132, wherein the lessor is provided with a code module in accordance with claim 49.

136. The method of claim 135, further comprising the lessor sending a code module to the remote device is response to receiving a registration request from the remote device.

137. The method of claim 123, further comprising the lessor receiving a request from the remote device to lease the file content to other devices.

138. The method of claim 137, further comprising the step of the lessor sending a code module in accordance with claim 49, to the other devices.

139. A host for executing an operation requested by a requestor, the host comprising a memory configured to store information identifying the requestor and information related to the operation requested by the requestor, and processing circuit for determining the presence of the requestor and for freeing resources associated with that operation if the requestor is not present.

140. A method of performing an operation in a host in response to a request from a requestor, the method comprising:

the host receiving a request from the requestor to perform an operation at the host;

the host storing information corresponding to the identity of the requestor at the host in the form of an object; and

the host freeing resources associated with that object if a refresh signal is not received by said host after a predetermined time.